УДК 004.91:378.147
DOI: 10.17223/16095944/69/2

Ю.П. Москалева, З.С. Сейдаметова
ГБОУВО РК «Крымский инженерно-педагогический университет»,
г. Симферополь, Россия

ОБУЧЕНИЕ СТУДЕНТОВ КОМАНДНОЙ РАБОТЕ С ПОМОЩЬЮ СИСТЕМ КОНТРОЛЯ ВЕРСИЙ

Умение использовать системы контроля версий (VCS) входит в перечень специальных навыков, необходимых выпускникам компьютерных специальностей для успешной работы в IT-отрасли. В связи с этим использование различных продуктов VCS в качестве учебной платформы является актуальным в учебном процессе современных университетов. В статье представлена методика обучения студентов компьютерных специальностей навыкам командной работы с помощью систем контроля версий (VCS), также проанализированы системы Subversion, Git и GitHub и представлены сценарии изучения централизованной и распределенной VCS.

Ключевые слова: командная работа, система контроля версий (VCS), программная инженерия, управление учебным курсом, централизованная VCS, распределенная VCS, Git, GitHub, Subversion.

Программные приложения – это огромное число строчек кода, которые создаются с помощью редакторов кода, интегрированных сред разработки (IDE). Успешная разработка приложений сопровождается созданием документации, подверженной постоянным и неконтролируемым изменениям, которые вносятся множеством участников проекта. За время разработки проект претерпевает изменения как в направлении кодирования новых частей и модулей, так и в создании новых версий существующих частей. Работа над проектом должна отслеживаться и сам проект модифицироваться таким образом, чтобы команда разработчиков координировала свои действия и имела четкое представление о том, что происходит с версиями приложения, какие части изменены. Для управления версиями и конфигурациями имеется достаточное количество инструментов, доступных как для платного, так и бесплатного пользования. Такой инструментарий называется системой контроля версий (Version Control System – VCS). Он позволяет разработчикам регистрировать изменения в одном или нескольких файлах, а также предоставлять возможность возвращаться к предыдущим версиям этих файлов.

Системы контроля (или управления) версиями (VCS) являются важными инструментами для разработчиков программных продуктов. Для формирования понимания реалий современной IT-индустрии у студентов компьютерных специальностей необходимо знакомить их с VCS, а также вырабатывать у студентов соответствующие навыки работы с системами контроля версий, такими, как, например, Subversion, Git и GitHub.

Системы управления исходным кодом имеют большое значение в программной инженерии. В статье Вижиса Ассара «Программное обеспечение, которое строит программное обеспечение» [1], опубликованной в журнале «The New Yorker», отмечено, что доминирующим игроком в разработке приложений стала быстрорастущая компания GitHub, которая полностью настроила инструменты контроля версий. Как отметил автор, сайт GitHub позволил программистам легко взаимодействовать через Интернет, предоставляя общение и социальные функции, например, следить за определенными кусками кода в проектах.

В статье И. Бонакдарьяна [2] описана структура подхода к разработке методических материалов и учебных ресурсов, позволяющих использовать VCS при обучении студентов компьютерных специальностей. Автор предлагает в этом подходе методику, состоящую из трех последовательных фаз: работа в командной строке, использование Git на локальном компьютере, работа с удаленным GitHub репозиторием. В статье Л. Хааранена и Т. Лехтинена [3] представлен опыт использования VCS, в частности, системы Git в учебных дисциплинах факультета компьютерных наук Университета Аалто в Финляндии. Авторы статьи [4] полагают, что Git и другие системы контроля версий трудны для изучения и использования. В статье [4] подробно описан клиент для Git – Elegit,

цель которого – помочь студентам понять, как работает система Git, и освоить работу с Git.

В статьях [5, 6] представлены результаты двух исследований использования GitHub в качестве платформы для организации студенческого сотрудничества в учебных IT-проектах. Авторы этих исследований сделали вывод, что использование GitHub в качестве платформы обучения, а также инструменты GitHub могут принести в университетское образование больше сотрудничества и прозрачности.

Структура локальных и удаленных репозиториев Git проанализирована в статьях [7–9]. В статье [8] П. Кочхар и Д. Ло проанализировали Git-репозитории Java-проектов, использующие assert-классы для выявления дефектов программы.

Цель настоящей статьи – исследовать концепцию контроля версий в контексте разработки программного приложения, а также представить методику обучения студентов компьютерных специальностей командной работе с использованием VCS Subversion, Git и GitHub.

Система контроля версий представляет собой приложение, позволяющее отслеживать изменения в файлах, хранить несколько версий файла, возвращаться к более ранним версиям, а также сохранять всю историю изменений. Контроль версий является частью конфигурационного управления.

VCS можно подразделить на три типа: локальные (используется локальный подход, все разработчики используют одинаковую файловую систему), централизованные (клиент-серверная модель, разработчики используют общий репозиторий) и распределенные (каждый разработчик работает непосредственно с собственным локальным репозиторием, а изменения распределяются между репозиториями отдельным шагом).

1. *Локальная система контроля версий* – это самая простая система. Локальная VCS позволяет сохранять различные версии программного обеспечения на локальной машине. Недостаток этого подхода – пользователь должен сам отслеживать все версии. Пользователь может внести изменения в неправильный каталог. Если пользователю будет нужно сотрудничать с другими разработчиками, то это невыполнимо в случае локальной VCS.

2. *Централизованные системы VCS* спроектированы таким образом: имеется один-единственный ресурс, куда обращаются разработчики. Все разработчики могут работать в этом ресурсе, добавлять изменения в коды. Примерами централизованных систем контроля версий являются Subversion, ClearCase, Perforce, VisualSourceSafe; они отличаются рабочими процессами, производительностью и интеграцией. Недостаток централизованных VCS – существует единственная точка отказа. В случае внештатной ситуации на сервере возможна катастрофическая потеря всего проекта.

3. *Распределенные системы VCS* (DVCS) разработаны таким образом, чтобы можно было работать в разных ресурсах и репозиториях, а проекты можно было передавать из одного хранилища в другое. Проектирование таких систем подразумевает различные формы связи. Любое семантическое значение, позволяющее определить, следует ли доверять хранилищу, навязывается извне процессом, а не самим программным обеспечением. Вся система в случае DVCS работает над «зеркалированием» репозиториев: весь код копируется всякий раз, когда мы выбираем клонирование некоторого репозитория, доступного на DVCS.

В табл. 1 представлена информация о некоторых системах контроля версий. Например, RCS имеет локальную модель репозитория и является одной из самых первых систем управления версиями. Первый релиз появился в 1982 г. Позволяет хранить историю изменений файла, но не имеет инструментов для коллективной работы. Системы с централизованной (клиент-серверной) моделью репозитория, представленные в табл. 1, – это ClearCase, CVS, Endevor, Rational Team Concert и Subversion. Примерами распределенной модели репозитория являются системы BitKeeper, Git, Mercurial.

В качестве учебной платформы можно выбирать как централизованные, так и распределенные системы контроля. В нашем случае мы предложили студентам для их проектов использовать Subversion и Git/GitHub. Эти два продукта представляют противоположные принципы проектирования в мире контроля версий. Subversion (SVN) следует клиент-серверной архитектуре с жестко определенными рабочими процессами. Контроль версий Git/GitHub имеет распределенную архитектуру со специальными рабочими процессами. Мы полагаем, что для понимания реальностей промышленной разработки программ-

Таблица 1

Общие и технические сведения о некоторых системах контроля версиями

Приложение	Модель согласования	Лицензия	Поддерживаемые платформы
Модель репозитория: локальная			
RCS (Revision Control System)	Слияние или блокировка	GNU GPL	Unix-подобные
Модель репозитория: централизованная (клиент-серверная)			
ClearCase	Слияние или блокировка	Проприетарная	Linux, Windows, AIX, Solaris, HP UX, i5/OS, OS/390, z/OS
CVS (Concurrent Versions System)	Слияние	GNU GPL	Unix-подобные, Windows, OS X
Endevor	Слияние или блокировка	Проприетарная	z/OS
Rational Team Concert	Слияние или блокировка	Проприетарная	Linux, Windows, AIX, Solaris, HP UX, i5/OS, OS/390, z/OS
Subversion (SVN)	Слияние или блокировка	Apache	Unix-подобные, Windows, OS X
Модель репозитория: распределенная			
BitKeeper	Слияние	Apache	Unix-подобные, Windows, OS X
Git	Слияние	GNU GPL	Unix-подобные, Windows, OS X
Mercurial	Слияние	GNU GPL	Unix-подобные, Windows, OS X

ного *обеспечения важно знакомить студентов с системами контроля версий, имеющих разную методологию*.

В табл. 2 представлены некоторые преимущества использования систем контроля: версии Subversion и Git/GitHub.

Для отработки навыков работы с централизованной VCS Subversion необходимо студентам предлагать следующие сценарии работы:

– С единственным редактированием. Проверка (check out), редактирование файла, передача (commit).

– С множественным редактированием. Управление изменением списка через несколько файлов, включая дополнения и удаления.

– С параллельным редактированием. Одновременное редактирование двух документов или файлов.

Деятельность в GitHub происходит в виде транзакций между «репозиториями», которые, по существу, являются справочниками кода. Хранилища могут быть клонированы, изменены, а затем снова объединены, поэтому программисты могут совместно создавать программное обеспечение и экспериментировать с ним, не усложняя процесс разработки.

Для работы с Git/GitHub (режим сотрудничества нескольких пользователей) на занятии предлагаются следующие сценарии:

– Разрешение конфликтов. Обработка коллизий и три варианта слияний.

– История. Выявление ответственного за частичные изменения и обучение тому, как через историю репозитория найти ответы на вопросы.

– Откат изменений. Отмена изменений, сделанных другим участником.

На рис. 1 представлен поток данных и доступ студента к материалам учебного курса в репозиториях Git. Задания составлены таким образом, чтобы студенты могли отработать следующие кон-

Таблица 2

Некоторые преимущества систем контроля, версии Subversion и Git/GitHub

Subversion	Git/GitHub
1. Номера глобальных ревизий – последовательные. Ревизия характеризует состояние не отдельного файла, а хранилища в целом. Это позволяет легко обнаруживать и анализировать ревизии. 2. Возможность проверить поддеревья. 3. svn:externals – свойство, которое позволяет части других репозиториев автоматически проверяться в других подкаталогах	1. Легко проводится локальное ветвление. Это позволяет работать над несколькими файлами / проектами параллельно, не допуская ошибок. 2. Работает быстрее Subversion, поскольку меньше обращается к серверу (в основном при выполнении git push или git pull). 3. Имеются инструменты для работы, например git bisect, которые упрощают отладку некоторых неочевидных проблем. 4. Возможность видеть всю история дерева, что позволяет работать без подключения к Интернету

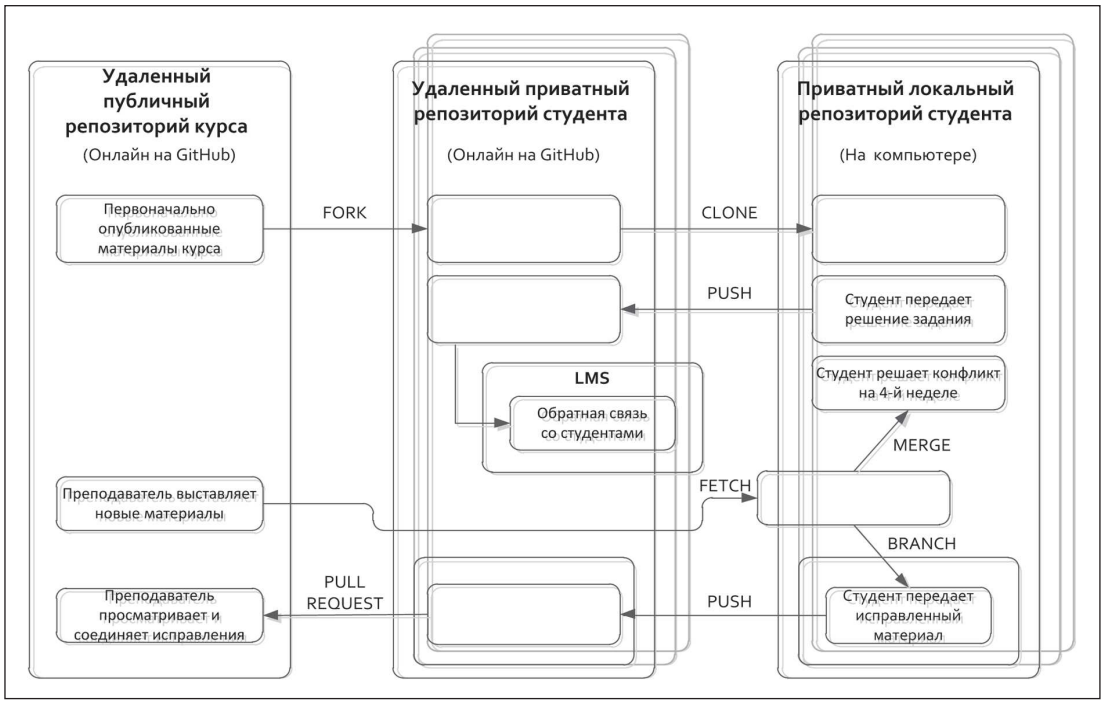

Рис. 1. Поток данных и доступ студента к учебным репозиториям Git

цепции: local & remote repository, fork, clone, add & commit, push, merge, branch, merge conflicts, pull request.

По нашему мнению, алгоритм обучения Git/GitHub на начальном этапе должен быть таким:

1) Запустить новый проект.

2) Создать ветвь, называемую, например, develop. С этого момента студент работает здесь.

3) Создать ответвление от develop и начать работу с исправлениями или создавать новые свойства.

4) Когда патч (часть кода) готов, создать pull request для своей ветви develop и выполнить слияние (merge).

5) Когда студент начинает уверенно выполнять предыдущие шаги, провести операцию слияния (merge) develop в ветку master.

6) Обозначить проект версией 0.1.0 и выпустить его (release).

7) Повторить, начиная с шага 3.

Этот алгоритм позволит научить студентов понятиям ветвления, управления версиями и создать навыки использования GitHub для совместной работы.

Таким образом, мы представили описание того, как системы контроля версиями Subversion, Git и GitHub могут использоваться в образовательном контексте; список преимуществ, предоставляемых хостингом и социальными функциями GitHub в образовании; представили рекомендации и предлагаемые методы использования Subversion, Git и GitHub для поддержки обучения. Следует отметить, что имеются проблемы и трудности, связанные с использованием Subversion, Git и GitHub в учебных курсах. Освоение VCS в учебном процессе требует существенного пересмотра методик преподавания многих дисциплин компьютерных специальностей и выстраивания новых связей между ними. В дальнейшем мы планируем проанализировать некоторые репозитории студентов, созданные ими в Git и GitHub для индивидуальных и групповых проектов.

ЛИТЕРАТУРА

1. *Assar V.* The software that builds software // The New Yorker. – 2013. – 07 avg. [Электронный ресурс]. – Режим доступа: http://bit.ly/2mHpHtF

2. *Bonakdarian E.* Pushing Git & GitHub in undergraduate computer science classes // Journal of Computing Sciences in Colleges. – January 2017. – Vol. 32, Issue 3. – USA: Consortium for Computing Sciences in Colleges, 2017. – P. 119–125.

3. *Haaranen L., Lehtinen T.* Teaching Git on the Side: Version Control System as a Course Platform // Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer

Science Education (June 4–8, 2015, Vilnius, Lithuania). – NY, USA: ACM, 2015. – P. 87–92.

4. *Walker E., Connelly J., Musicant D.* Elegit: Git Learning Tool for Students // Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education Seattle (March 8–11, 2017. – Washington, USA). – NY, USA: ACM, 2017. P. 642.

5. *Zagalsky A., Feliciano J., Storey M.-A. et al.* The Emergence of GitHub as a Collaborative Platform for Education // Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing (Vancouver, BC, Canada. March 14–18, 2015). – NY, USA: ACM, 2015. – P. 1906–1917.

6. *Fontana F.A., Raibulet C.* Students' Feedback in Using GitHub in a Project Development for a Software Engineering Course // Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education (Bologna, Italy. July 03–05, 2017). – NY, USA: ACM, 2017. – P. 1906–1917.

7. *Zhu J., Zhou M., Mockus A.* Patterns of folder use and project popularity: a case study of GitHub repositories // Proceedings of the 8th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (Torino, Italy. September 18–19, 2014. Article № 30). – NY, USA: ACM, 2014. – 4 p.

8. *Kochhar P.S., Lo D.* Revisiting Assert Use in GitHub Projects // Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering (Karlskrona, Sweden. June 15–16, 2017). – NY, USA: ACM, 2017. – P. 298–307.

9. *Borges H., Hora A., Valente M.T.* Predicting the Popularity of GitHub Repositories // Proceedings of the 12th International Conference on Predictive Models and Data Analytics in Software Engineering (Ciudad Real, Spain. September 09, 2016. Article № 9). – NY, USA: ACM, 2016. – 10 p.

Moskaleva Y.P., Seidametova Z.S.
Crimean Engineering-Pedagogical University, Simferopol, Russia

TEACHING TEAMWORK USES SYSTEM CONTROL VERSIONS

Keywords: teamwork, system control version (VCS), software engineering, centralized VCS, distributed VCS, Git, GitHub, Subversion.

The development of the software applications is accompanied by the creation of different types of documentation, which is subjected to constant and uncontrolled changes. These changes are made by a large numbers of the project participants. The team of developers should coordinate their teamwork in the project so that to clearly understand what happens to the software application versions and which parts of the program are changed. There are a sufficient numbers of tools that are available for both commercial and freeware using for managing of the software application versions and configurations. The type of the software tools is called the Version Control Systems (VCS). Systems like VCS allow developers to register changes in one or more files, as well as provide an opportunity to return to the previous versions of these files.

Version Control Systems are important tools for the software developers. As a university teacher we must familiarize computer science students with VCS for the formation of understanding of the modern IT-industry realities. We also must develop appropriate skills in working with Version Control Systems, for example, VCS like the centralized system Subversion, and distributed systems Git and GitHub.

The skills of using the version control systems are included into the list of special competences required for undergraduates and graduates in computer science majors for the successful work in the computing area. In this regard, the use of various VCS products as a training platform is relevant in the educational process of the universities. The article presents a methodology for teaching students to computer science, their teamwork skills via the version control systems such as Subversion, Git and GitHub. We have also analyzed different VCS and presented the scenarios for teaching centralized and distributed VCS.

We present the algorithm that will help students to understand the concepts of branching, version control and will create skills for using GitHub for team-working.

It is important to understand how Subversion, Git and GitHub version control systems can be used in teaching and learning context. We describe the list of benefits provided by the hosting and social functions of GitHub for education. We also present recommendations and methods for using Subversion, Git and GitHub in learning support.

There are some problems and difficulties associated with the Subversion, Git and GitHub using in courses related to programming and software development: the learning of the VCS in undergraduate and graduate studies requires a substantial revision of the teaching methods of many computer science courses and the building of new ties between them.

REFERENCES

1. *Assar V.* The software that builds software // The New Yorker. – 2013. – 07 avg. [Jelektronnyj resurs]. – Rezhim dostupa: http://bit.ly/2mHpHtF

2. *Bonakdarian E.* Pushing Git & GitHub in undergraduate computer science classes // Journal of Computing Sciences in

Colleges. – January 2017. – Vol. 32, Issue 3. – USA: Consortium for Computing Sciences in Colleges, 2017. – P. 119–125.

3. *Haaranen L., Lehtinen T.* Teaching Git on the Side: Version Control System as a Course Platform // Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education (June 4–8, 2015, Vilnius, Lithuania). – NY, USA: ACM, 2015. – P. 87–92.

4. *Walker E., Connelly J., Musicant D.* Elegit: Git Learning Tool for Students // Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education Seattle (March 8–11, 2017, Washington, USA). – NY, USA: ACM, 2017. – P. 642.

5. *Zagalsky A., Feliciano J., Storey M.-A. et al.* The Emergence of GitHub as a Collaborative Platform for Education // Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing (Vancouver, BC, Canada. March 14–18, 2015). – NY, USA: ACM, 2015. – P. 1906–1917.

6. *Fontana F.A., Raibulet C.* Students' Feedback in Using GitHub in a Project Development for a Software Engineering Course // Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education (Bologna, Italy. July 03–05, 2017). – NY, USA: ACM, 2017. – P. 1906–1917.

7. *Zhu J., Zhou M., Mockus A.* Patterns of folder use and project popularity: a case study of GitHub repositories // Proceedings of the 8th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (Torino, Italy. September 18–19, 2014. Article № 30). – NY, USA: ACM, 2014. – 4 p.

8. *Kochhar P.S., Lo D.* Revisiting Assert Use in GitHub Projects // Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering (Karlskrona, Sweden. June 15–16, 2017). – NY, USA: ACM, 2017. – P. 298–307.

9. *Borges H., Hora A., Valente M.T.* Predicting the Popularity of GitHub Repositories // Proceedings of the 12th International Conference on Predictive Models and Data Analytics in Software Engineering (Ciudad Real, Spain. September 09, 2016. Article № 9). – NY, USA: ACM, 2016. – 10 p.